APPENDIX A

Drive Forward Experiment

```
void setup()
{
  motors.drive(255);   // Turn on Left and right motors at full speed forward.
  delay(2000);         // Waits for 2 seconds
  motors.stop();       // Stops both motors
}

void loop()
{
  // Nothing here.
}
```

[motors.drive([motorPower]) turns both motors. This method takes one input parameter, [motorPower]. [motorPower] can be any integer value from -255 to +255. Values > 0 will cause the robot to drive forward -- spinning the right motor clockwise (CW) and the left motor counter-clockwise (CCW) -- driving the robot forward. Values < 0 will do the opposite causing the robot to drive in reverse.

motors.drive(255); // drives forward at full power. motors.drive(-255); // drives reverse at full power. [STUDENT DOES=NEW] Measure the distance your robot travels in 2 seconds. [TEACHER NOTES: 110 cm in 2 secs]
Sometimes running the motors at full power causes the wheels to spin-out. If you notice traction issues, you may want to try playing around with slower speeds.

Stopping motors.stop(); turns off power to both motors and coasts to a stop.

motors.stop(); // sets the motor speeds to 0 and coasts to a stop.
Sometimes, you might want a more precise stop. the RedBotMotors class also has a brake() method that forces the motors to come to a more abrupt stop.

Try replacing the motors.stop(); with motors.brake(); in your example code. [STUDENT DOES=NEW] Measure the distance your robot travels.

Positive speeds cause the RedBot to go forward by spinning the right side clockwise and the left side counter-clockwise. To control the individual motors, the RedBotMotors class has two methods that are used.

```
motors.rightMotor([motorPower]);  // controls the right motor
motors.leftMotor([motorPower]);   // controls the left motor
```

Similar to the .drive([motorPower]) method, [motorPower] values can vary from -255 to +255. Positive values spin the motor in the clockwise direction and Negative values spin the motor in the counterclockwise direction.

Turning Corners Experiment:

You may need to adjust the speeds and the delay() times to get a good angle. We can write a sub-routine called turnAngle(). This function will use the average turningSpeed you calculated. ***Take the motorPower that you used and the turningTime to calculate the "turningSpeed" of your RedBot. We define turningSpeed = angle / time. In this case, it should be 90 degrees divided by the turningTime you used. Write down your turningSpeed.***

Copy and paste this block of code to the end of your program -- after the void loop(){}

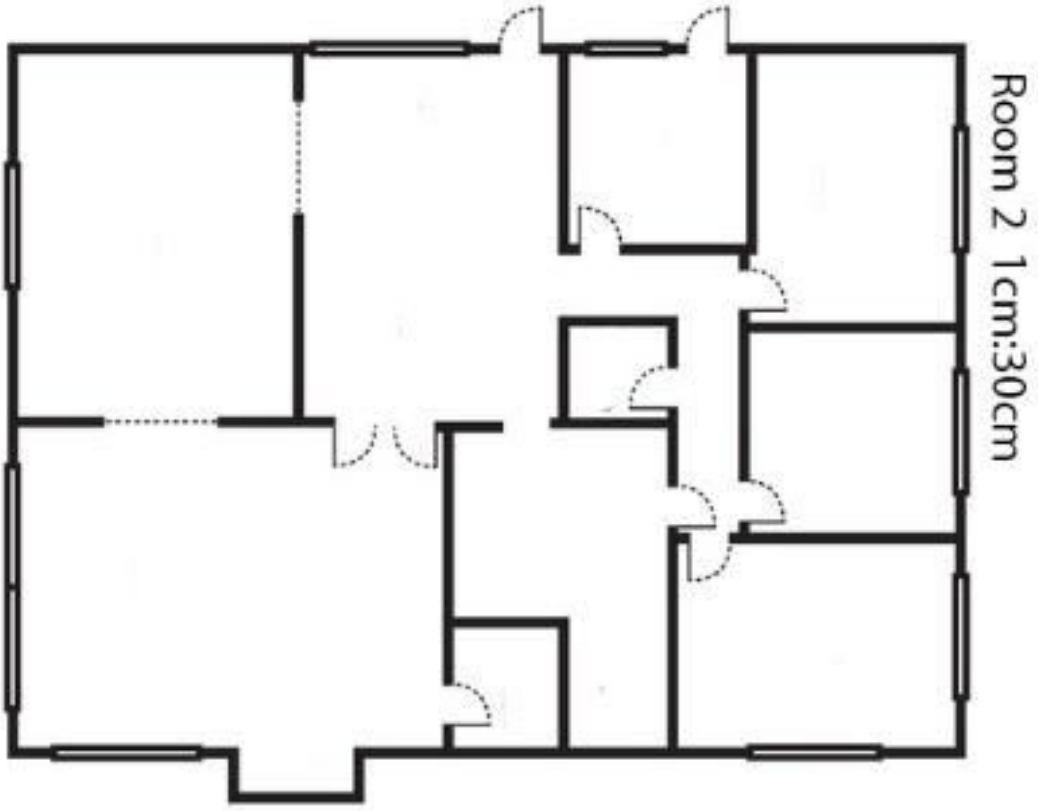
```
void turnAngle(int angle)
{
 int turningSpeed = 180; // degreees / second

 long turningTime;
 turningTime = (long) 1000 * angle / turningSpeed;

 motors.rightMotor(-100); // Turn CCW at motorPower of 100
 motors.leftMotor(-100);  // Turn CCW at motorPower of 100
 delay(turningTime);      // turning Time
 motors.brake();     // brake() motors
}
```

Now, replace your turning code with a function call turnAngle(90); <u>Upload and test.</u> Your RedBot should still be turning 90 degrees. If it's turning too far or not far enough, then adjust the variable turningSpeed until your RedBot is turning a good 90 degrees. You can now change the turning angle as needed. Use this with the driveDistance() function to travel a distance.

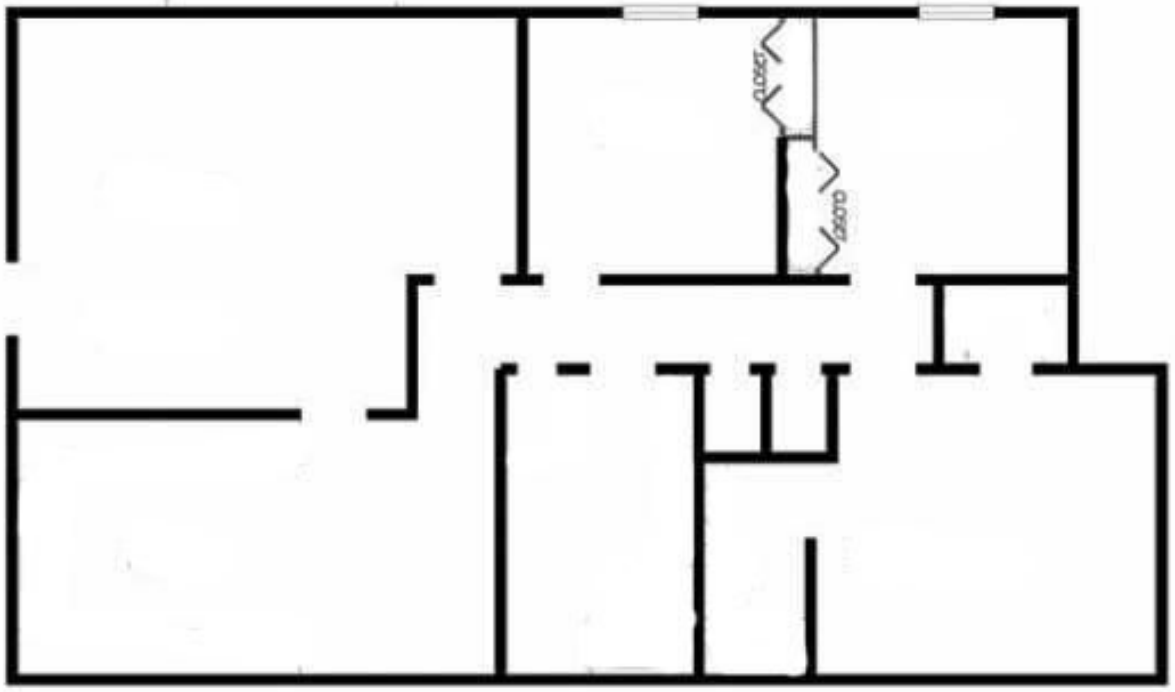
The RedBot is pretty nimble, so we want to use a lower motorPowers for pivoting. In our example, we set the motorPower to 100 and the delay time to 0.5 seconds. Play around with this until you're able to get your robot to turn a nice 90 degree turn consistently.

[TEACHER QUESTIONS: Too abrupt of a turn? How can I turn more gradually? Play around with varying the power to the right side vs. the left side.]

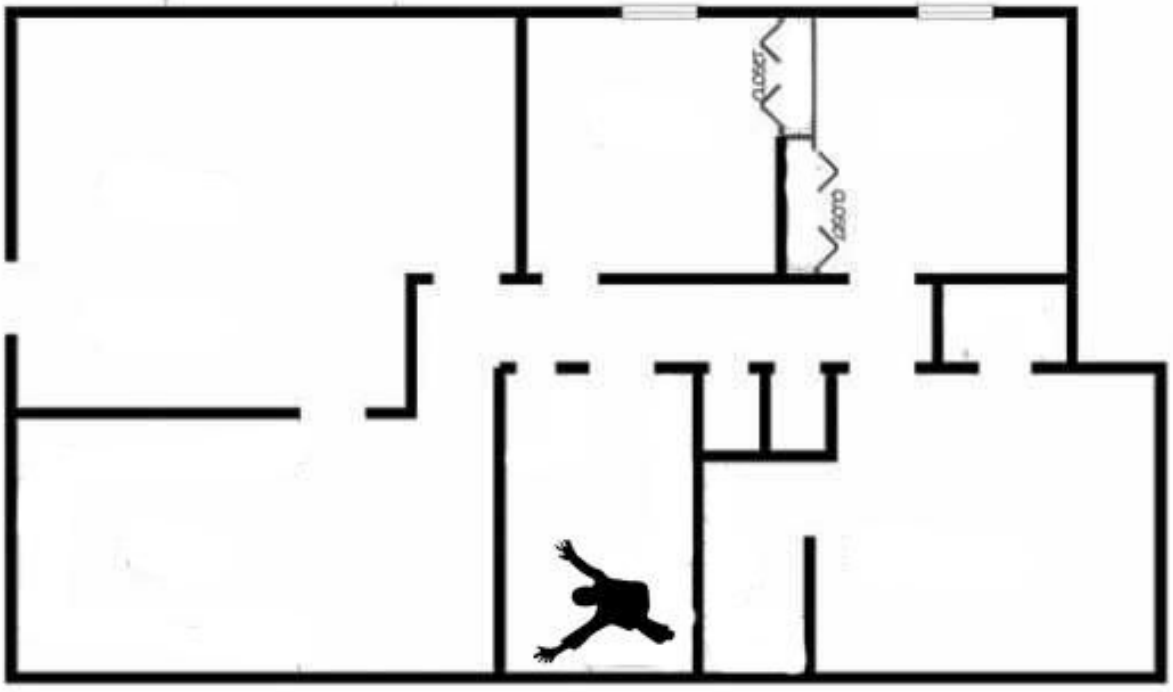
APPENDIX B



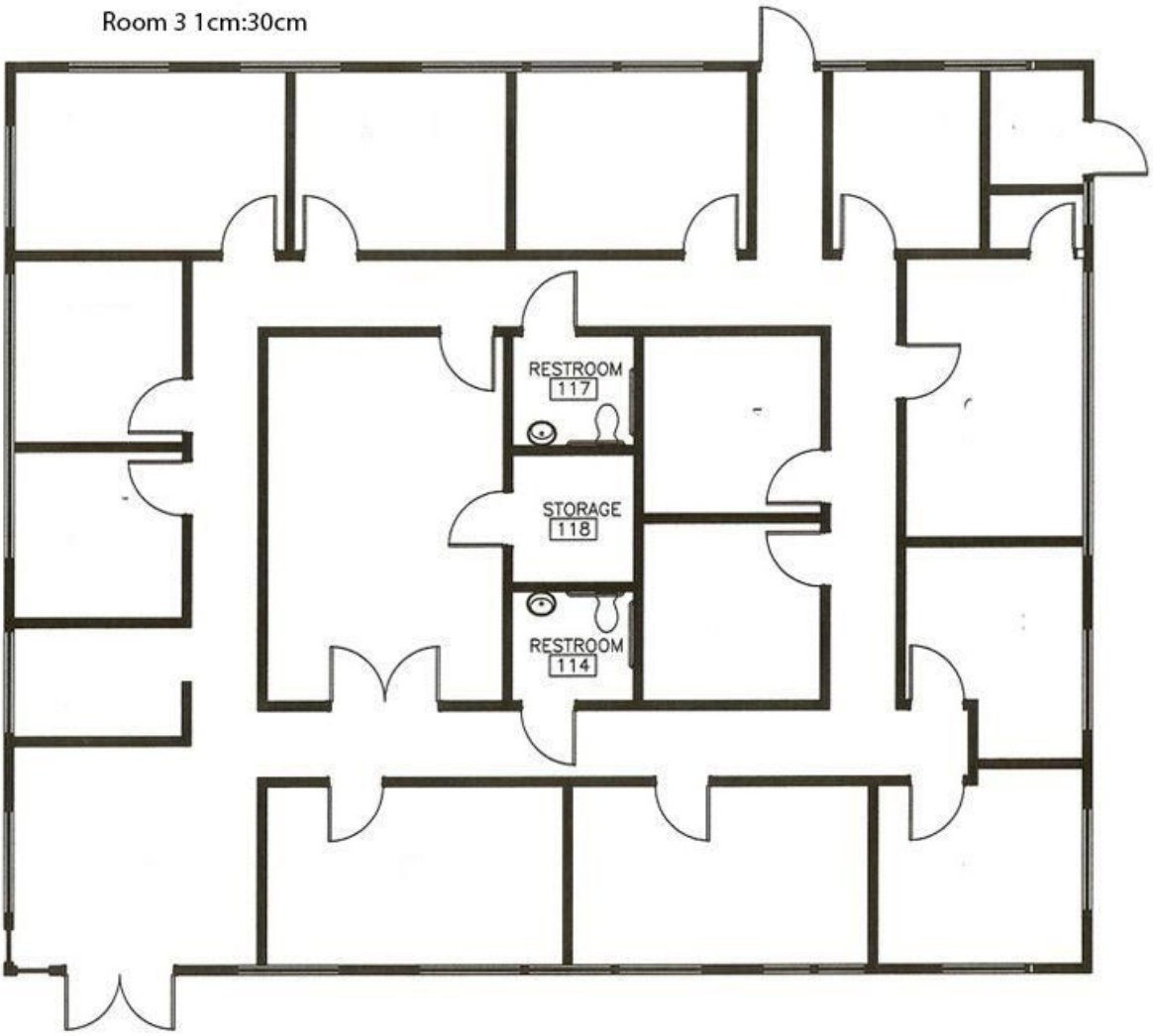
Room 1 1cm:20cm



Room 1 1cm:20cm SCENARIO 1



Room 3 1cm:30cm



Appendix C

1. **Run a minimum of 5 trials of the Drive Forward with your robot and measure how far your robot travels in 2 seconds.**

Trials	**Distance Traveled**	**Time traveled**
1		
2		
3		
4		
5		
Average		

2. **Calculate the average speed of your robot (in inches per second) using the equation: avgSpeed = distance / time.**

Route (steps) command	Scale Drawing Measurements	Measurements needed for actual drawing	Distance converted to time	Code language
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				